**Amendments to the Specification:**

Please replace paragraph [0006] with the following amended paragraph:

**[0006]** Another approach for the elimination of dead code is based on extending the single live bit to a set of bits based on the computation being performed with a specific number of bits as disclosed in U.S. Patent No. 6,571,387 entitled "Method and computer program product for global minimization of sign-extension and zero-extension operations. The number of bits may not be supported ~~be~~ by a host processor, such as a host processor might require computations to be done in sixty-four (64) bit integers, while the program instructions have been specific to thirty-two (32) bits integers. In this approach, a compiler may insert instructions to compensate for this difference. If a compiler can determine that all of the upper bits are dead code, the extra inserted instructions may be deleted.

Please replace paragraph [0008] with the following amended paragraph:

**[0008]** In a single instruction multiple data (SIMD) processing environment, there are advantages to using a superword register, wherein a superword register includes a hardware resource that can hold a small, but more than one, number of words of data. In one exemplary orientation, the superword register can hold up to 128 bits divided into four floating point components.

Please replace paragraph [0009] with the following amended paragraph:

**[0009]** Instructions that operate on superword registers operate in parallel on all components and therefore are capable of achieving very high performance provided that more than one component contains data. If a program computes an unused value in a superword

component, then the value of the component is said to be dead, whereas the other ~~elements~~ components are termed live. Some compilers using superword registers fail to eliminate the dead code and therefore reduce operating efficiency due to extra un-needed components. For example, the disclosure of U.S. Patent No. 6,571,387 does not provide for dead code elimination using a superword register. Superword registers provide improved processing times in a SIMD environment and there does not exist a dead code elimination technique with a superword register. Therefore, there exists a need for removing dead code from a superword register.

Please replace paragraph [0015] with the following amended paragraph:

**[0015]** Briefly, a method and apparatus for dead code elimination includes adding to each instruction ~~(1) a set~~(1) a set of bits called a write mask (one per component) used to indicate the components produced by the instruction, (2) a link called the "previous write" linking the instruction with the prior instruction writing components of the same superword, and (3) a set of bits (one per component) called live bits, indicating the components consumed by subsequent uses of the superword register and the instruction is in SSA form. Initially all live bits are set to false.

Please replace paragraph [0017] with the following amended paragraph:

**[0017]** As each instruction I is removed from the worklist, a backward walk over the ssa links and partial write links, can be used to identify each instruction J which ~~produce~~ produces a source used by I. Based on the live bits in I, the corresponding live bits of J may be increased. If I uses a component produced by J and the live bits of J for that component are not set, then set the live bit for that component in instruction to on, and add J to the worklist.

Please replace paragraph [0019] with the following amended paragraph:

**[0019]** In the method and apparatus, each of the instructions include a previous link, a list of live components and a write mask, where a previous link connects instructions that write different components of the same resource and the write mask is an n-bit field where n is determined by the number of ~~elements~~ components of a superword register. For example, if a superword register allows for four components, this provides for a write mask having four bits, such as an instruction associated with an x, y, z and w field.

Please replace paragraph [0020] with the following amended paragraph:

**[0020]** The method and apparatus further includes determining if all ~~elements~~ components within a particular resource of the at least one second instruction are required ~~for the at least one second instruction~~. As discussed above, in one embodiment having 4 ~~elements~~components, a determination is made if the x field is required, if the y field is required, if the z field is required and if the w field is required. If ~~all~~ none of the components ~~the elements~~ are required, the method and apparatus provides for deleting the first instruction as it is determined that this instruction is extraneous, otherwise referred to as dead code. Therefore, the method and apparatus provides for elimination of dead code by examining the steps of the instructions in accordance with the parameters of the SSA based instructions.

Please replace paragraph [0021] with the following amended paragraph:

**[0021]** FIG. 1 illustrates the steps of one embodiment of a method for eliminating dead code. The method begins, step 100, by examining a first instruction off of a worklist, wherein

the first instruction includes a previous link, live bits and a write mask, step 102. A standard instruction includes a first term and a second term, wherein the instructions disposed on the worklist have the three additional fields added thereto. The previous link is a field that indicates a previous instruction and the write mask allows for operation and coordination of instructions with a superword register. The ~~live bits indicate~~write mask indicates components produced by this instruction or by a prior instruction ~~that can be reached~~ via the previous ~~write~~ link. As the superword register has a plurality of ~~elements~~components, the write mask coordinates instructions with specific components. This allows for operation of compiler operations such as a swizzle and maintenance of ~~element~~component-specific based calculations.

Please replace paragraph [0022] with the following amended paragraph:

**[0022]** The first instruction is examined to determine its source instructions. ~~if any of the elements are live for a particular calculation. If a particular element is not live, this indicates that this instruction is dead code and should be eliminated from the generated machine code.~~ Step 104 is examining ~~one or more~~at least one second ~~instructions~~instruction, wherein ~~the one or more second instructions are~~ each is a source ~~instructions~~ instruction of the first instruction and each ~~of the at least one second instructions include~~ includes a previous link, a set of live bits and a write mask. The previous link, live bits and the write mask of the ~~one or more~~at least one second ~~instructions have~~ instruction has the same structure of the previous link, live bits and the write mask as the first instruction pulled off of the worklist.

Please replace paragraph [0023] with the following amended paragraph:

**[0023]** Step 106 is determining if ~~all elements~~any components within a ~~particular~~ field of the at least one second instruction are required ~~for the at least one second instruction~~. This step is performed by looking at each ~~element~~component. In the embodiment having four ~~elements~~component, the x ~~element~~ component is examined, the y ~~element~~ component is examined, the z ~~element~~ component is examined and the w ~~element~~ component is examined to determine if ~~the element is~~ each component is live (i.e., required). Also noting in the method and apparatus that with the operation of a swizzle function, the various ~~elements~~ components may be representative of other specified ~~elements~~components.

Please replace paragraph [0024] with the following amended paragraph:

**[0024]** Step 108 is deleting the ~~first~~ second instruction from the machine code if no components are live. If no components are live, this means that this particular instruction is extraneous because result of this instruction ~~are~~ is not used by another instruction. Thereupon, the method is complete, step 110.

Please replace paragraph [0028] with the following amended paragraph:

**[0028]** Step 134 is examining further instructions, the at least one second instruction ~~off of the worklist~~. This method is an iterative process looking at multiple instructions to determine which instructions are extraneous, dead code. Therefore, step 136 is determining if ~~all~~ any component ~~elements~~ within a particular field of the at least one second instruction ~~are~~ is required for the ~~subsequent~~ first instruction ~~instructions, the at least one second instructions~~. If no ~~elements~~ components are required, step 138 is deleting the ~~first~~ second instruction from the

machine program, as this instruction is dead code. If any component is required, the second instruction is added to the worklist, step 139. Thereupon, the method is complete, step 140.

Please replace paragraph [0034] with the following amended paragraph:

**[0034]** FIG. 4 illustrates a graphical representation of an instruction 170, in accordance with one embodiment of the present invention. The instruction includes the two ~~elements~~ components of the instruction, S1 172 and S2 174. Although, in the present invention the previous link field 176 is added, as well as the write mask 178. The added write mask 178 allows for usage of the instruction with the superword register based on tracking the superword register components. It should also be noted that in another embodiment, the instruction further includes a live bit, as discussed above.

Please replace paragraph [0037] with the following amended paragraph:

**[0037]** In the elimination of dead code and the instructions illustrated in FIG. 5 having SSA form, the worklist is utilized to identify steps that are dead code. Therefore, in examining export 2 202, this is the result of 1B 240. 1B 240 is the result of an operation with 2C 242 and 2D 248. Walking up the SSA links, ~~element~~ instruction 2C 242 is the result of 3B 222 and 3C 244. As 3B does not have any higher instructions, step 3C is examined as the result of 4C 246. Back down the SSA links, the second ~~element~~ instruction for 1B 240 was 2D 248, which is the result of 3D 250, wherein 3D is the result of 4C 246 and 4D 252.

Please replace paragraph [0038] with the following amended paragraph:

**[0038]** Following the progression of the third export 204, 1C 260 is based on 2D 248 and 2E 262. The instructions for 2D 248 were previously determined with respect to the second export 202, therefore, the ~~steps~~ instructions above 2D may be removed as dead code using the previous ~~bit~~link, the write mask and in one embodiment the live bit for the SSA form instructions. With respect to 2E 262, this is the result of 3E 264 which is the result of 4D 252 and 4E 266. In the event there are any further instructions above 4D 252, these would have been determined upon inspection of the second export 202, therefore these instructions could also be eliminated as dead code.